BEST AVAILABLE COPY

The class java.lang.Class

```
package java.lang;

import java.lang.reflect.Field;

import java.lang.reflect.Method;

import java.lang.reflect.Constructor;

public final class Class extends Object
```

Instances of the class `Class` represent Java types in a way that allows them to be manipulated by a running Java program. Every array also belongs to a class that is reflected as a `Class` object that is shared by all arrays with the same element type and number of dimensions. Finally, the eight primitive Java types and `void` are also represented by unique `Class` objects.

There are no public constructors for class `Class`. The Java Virtual Machine automatically constructs `Class` objects when new classes are loaded; such objects cannot be created by user programs.

While the class `Class` properly belongs in the `java.lang.reflect` package, it remains in `java.lang` for backwards compatibility.

Methods

The class `Class` is augmented with new methods to:

- determine if a `Class` object represents an array type
- determine if a `Class` object represents a primitive type
- determine the Java language modifiers of the represented class type
- reflect the members and constructors of the represented type
- determine if an object is an instance of the represented class, or implements the represented interface
- determine if a class or interface is a superclass or superinterface of a class or interface
- get the component type for a represented array type

The existing and new methods of class `Class` are described below.

toString

```
public String toString()
```

Returns a `String` consisting of the word `class`, a space, and the fully-qualified name of the class if this `Class` object represents a class (either a declared class or an array class). If this `Class` object represents an interface, then this method returns a `String` consisting of the word `interface`, followed by a space, followed by the fully-qualified name of the interface. If this `Class` object represents a primitive type, then this method returns the name of the primitive type. If this `Class` object represents `void`, returns the String `"void"`.

This method overrides the `toString` method of class `Object`.

forName

```
public static Class forName(String className)

     throws ClassNotFoundException
```

Given the fully-qualified name for a class, this method attempts to locate, load, and link the specified class. If it succeeds, returns the `Class` object representing the class. If it fails, the method throws a `ClassNotFoundException`.

`Class` objects for array types may be obtained via this method. These Class objects are automatically constructed by the Java Virtual Machine.

`Class` objects that represent the primitive Java types or `void` cannot be obtained via this method.

newInstance

```
public Object newInstance()

     throws InstantiationException, IllegalAccessException
```

Creates and initializes a new instance of the class represented by this `Class` object, provided it represents an instantiable class. This is done exactly as if by an instance creation expression with an empty argument list. If evaluation of such an instance creation expression would complete abruptly, then the invocation of `newInstance` will complete abruptly for the same reason. Otherwise, it returns the newly created and initialized instance.

The method throws an `IllegalAccessException` if the class or initializer is not accessible to the calling class. The method throws an `InstantiationException` if it attempts to instantiate an abstract class or an interface, or if it is invoked on a `Class` object that represents a primitive type or `void`.

isInstance

```
public boolean isInstance(Object obj)
```

This method is the dynamic equivalent of the Java language `instanceof` operator. The method returns `true` if the specified `Object` argument is non- `null` and can be cast to the reference type represented by this `Class` object without raising a `ClassCastException`. It returns `false` otherwise.

If this `Class` object represents a primitive type or `void`, returns `false`.

See *The Java Language Specification*, section 15.19.2.

isAssignableFrom

```
public boolean isAssignableFrom(Class fromClass)
```

This method tests whether the type represented by the specified `Class` parameter can be converted to the type represented by this `Class` object via an identity conversion or via a widening reference conversion. It returns `true` if so, `false` otherwise.

If this `Class` object represents a primitive type, returns `true` if the specified `Class` parameter is exactly this `Class` object, `false` otherwise.

This method throws a `NullPointerException` if the specified `Class` parameter is `null`.

See *The Java Language Specification*, sections 5.1.1, 5.1.4 and 5.2.

isInterface

```
public boolean isInterface()
```

If this `Class` object represents an interface type, returns `true`, otherwise returns `false`.

isArray

```
public boolean isArray()
```

If this `Class` object represents an array type, returns `true`; otherwise returns `false`.

isPrimitive

```
public boolean isPrimitive()
```

If this `Class` object represents a primitive Java type, returns `true`; otherwise returns `false`.

There are nine predefined `Class` objects that represent theprimitive Java types and `void`. These are created by the Java Virtual Machine, and have the same names as the primitive types that they represent. These objects may only be accessed via the following `public static final` variables:

```
java.lang.Boolean.TYPE

java.lang.Character.TYPE

java.lang.Byte.TYPE

java.lang.Short.TYPE
java.lang.Integer.TYPE

java.lang.Long.TYPE

java.lang.Float.TYPE

java.lang.Double.TYPE

java.lang.Void.TYPE
```

These are the only `Class` objects for which this method returns `true`.

getName

```
public String getName()
```

Returns the fully-qualified name of the class (declared or array), interface, primitive type or `void` represented by this `Class` object, as a `String`.

getModifiers

```
public int getModifiers()
```

Returns the Java language modifiers for this class or interface, encoded in an integer. The modifiers consist of the Java Virtual Machine's constants for `public`, `protected`, `private`, `final`, and `interface`; they should be decoded using the methods of class `Modifier`.

The modifier encodings are defined in *The Java Virtual Machine Specification*, table 4.1.

getClassLoader

```
public ClassLoader getClassLoader()
```

Returns the class loader object that loaded this `Class`. Returns `null` if this `Class` was not loaded by a class loader.

getSuperclass

```
public Class getSuperclass()
```

If this `Class` object represents a class other than `Object`, returns the `Class` that represents the superclass of the class. Returns `null` if this `Class` represents the class `Object`, or if it represents an interface type or a primitive type.

getInterfaces

```
public Class[] getInterfaces()
```

Returns an array of `Class` objects representing the interfaces of the class or interface represented by this `Class` object. If this `Class` object represents a class, returns an array containing objects representing the interfaces directly implemented by the class. If this `Class` object represents an interface, returns an array containing objects representing the direct superinterfaces of the interface. Returns an array of length `0` if this `Class` implements no interfaces or if it represents a primitive type.

getComponentType

```
public Class getComponentType()
```

If this `Class` object represents an array type, returns the `Class` object representing the component type of the array; otherwise returns `null`.

getDeclaringClass

```
public Class getDeclaringClass()
```

If this class or interface is a member of another class, returns the `Class` object representing the class of which it is a member (its *declaring class*). Returns a `null` reference if this class or interface is not a member of any other class.

getClasses

```
public Class[] getClasses()
```

Returns an array containing `Class` objects representing all the `public` classes and interfaces that are members of the class represented by this `Class` object. This includes `public` class and interface members inherited from superclasses and `public` class and interface members declared by the class. Returns an array of length `0` if the class has no public member classes or interfaces, or if this `Class` object represents a primitive type.

getFields

```
public Field[] getFields()
    throws SecurityException
```

Returns an array containing `Field` objects reflecting all the `public` *accessible* fields of the class or interface represented by this `Class` object, including those declared by the class or interface and those declared by superclasses and superinterfaces. (Thus, the array includes the `public` *member* fields of the class as well as any additional `public` *hidden* fields.) Returns an array of length `0` if the class or interface has no `public` accessible fields.

Note that the implicit `length` field for array types is not reflected by this method. User code should use the methods of class `Array` to manipulate arrays.

The method throws a `SecurityException` if access to this information is denied.

See *The Java Language Specification*, sections 8.2 and 8.3.

getMethods

```
public Method[] getMethods()
```

```
        throws SecurityException
```

Returns an array containing `Method` objects reflecting all the `public` *member* methods of the class or interface represented by this `Class` object, including those declared by the class or interface and and those inherited from superclasses and superinterfaces. Returns an array of length `0` if the class or interface has no `public` member methods.

The method throws a `SecurityException` if access to this information is denied.

See *The Java Language Specification*, sections 8.2 and 8.4.

getConstructors

```
    public Constructor[] getConstructors()

        throws SecurityException
```

Returns an array containing `Constructor` objects that reflect all the `public` constructors of the class represented by this `Class` object. Returns an array of length `0` if the class has no `public` constructors, or if this `Class` object represents an interface or a primitive type.

The method throws a `SecurityException` if access to this information is denied.

getField

```
    public Field getField(String name)

        throws NoSuchFieldException, SecurityException
```

Returns a `Field` object that reflects the specified `public` *accessible* field of the class or interface represented by this `Class` object. The `name` parameter is a `String` specifying the simple name of the desired field.

The field to be reflected is located by searching all the accessible fields of the class or interface represented by this `Class` object for a `public` field with the specified name.

The method throws a `NoSuchFieldException` if a matching field is not found.

The method throws a `SecurityException` if access to the underlying field is denied.

See *The Java Language Specification*, sections 8.2 and 8.3.

getMethod

```
    public Method getMethod(String name, Class[] parameterTypes)

        throws NoSuchMethodException, SecurityException
```

Returns a `Method` object that reflects the specified public *member* method of the class or interface represented by this `Class` object. The `name` parameter is a `String` specifying the simple name the desired method, and the `parameterTypes` parameter is an array of `Class` objects that identify the method's formal parameter types, in declared order.

The method to reflect is located by searching all the member methods of the class or interface represented by this `Class` object for a `public` method with the specified name and exactly the same formal parameter types.

The method throws a `NoSuchMethodException` a matching method is not found.

The method throws a `SecurityException` if access to the underlying method is denied.

See *The Java Language Specification*, sections 8.2 and 8.4.

getConstructor

```
public Constructor getConstructor(Class[] parameterTypes)

    throws NoSuchMethodException, SecurityException
```

Returns a `Constructor` object that reflects the specified `public` constructor of the class represented by this `Class` object. The `parameterTypes` parameter is an array of `Class` objects that identify the constructor's formal parameter types, in declared order.

The constructor to reflect is located by searching all the constructors of the class represented by this `Class` object for a `public` constructor with the exactly the same formal parameter types.

The method throws a `NoSuchMethodException` if a matching constructor is not found.

The method throws a `SecurityException` if access to the underlying constructor is denied.

getDeclaredClasses

```
public Class[] getDeclaredClasses()

    throws SecurityException
```

Returns an array of `Class` objects reflecting all the classes and interfaces declared as members of the class represented by this `Class` object. This includes `public`, `protected`, default (package) access, and `private` classes and interfaces declared by the class, but excludes inherited classes and interfaces. Returns an array of length `0` if the class declares no classes or interfaces as members, or if this `Class` object represents a primitive type.

The method throws a `SecurityException` if access to this information is denied.

getDeclaredFields

```
public Field[] getDeclaredFields()

    throws SecurityException
```

Returns an array of `Field` objects reflecting all the fields declared by the class or interface represented by this `Class` object. This includes `public`, `protected`, default (package) access, and `private` fields, but excludes inherited fields. Returns an array of length `0` if the class or interface declares no fields, or if this `Class` object represents a primitive type.

The method throws a `SecurityException` if access to this information is denied.

See *The Java Language Specification*, sections 8.2 and 8.3.

getDeclaredMethods

```
    public Method[] getDeclaredMethods()

        throws SecurityException
```

Returns an array of `Method` objects reflecting all the methods declared by the class or interface represented by this `Class` object. This includes `public`, `protected`, default (package) access, and `private` methods, but excludes inherited methods. Returns an array of length `0` if the class or interface declares no methods, or if this `Class` object represents a primitive type.

The method throws a `SecurityException` if access to this information is denied.

See *The Java Language Specification*, sections 8.2 and 8.4.

getDeclaredConstructors

```
    public Constructor[] getDeclaredConstructors()

        throws SecurityException
```

Returns an array of `Constructor` objects reflecting all the constructors declared by the class represented by this `Class` object. These are `public`, `protected`, default (package) access, and `private` constructors. Returns an array of length `0` if this `Class` object represents an interface or a primitive type.

The method throws a `SecurityException` if access to this information is denied.

getDeclaredField

```
    public Field getDeclaredField(String name)

        throws NoSuchFieldException, SecurityException
```

Returns a `Field` object that reflects the specified declared field of the class or interface represented by this `Class` object. The `name` parameter is a `String` that specifies the simple name of the desired field.

The method throws a `NoSuchFieldException` if a field with the specified name is not found.

The method throws a `SecurityException` if access to the underlying field is denied.

See *The Java Language Specification*, sections 8.2 and 8.3.

getDeclaredMethod

```
    public Method getDeclaredMethod(String name,
                                    Class[] parameterTypes)
        throws NoSuchMethodException, SecurityException
```

Returns a `Method` object that reflects the specified declared method of the class or interface represented by this `Class` object. The `name` parameter is a `String` that specifies the simple name of the desired method, and the `parameterTypes` parameter is an array of `Class` objects that identify the method's formal parameter types, in declared order.

The method throws a `NoSuchMethodException` if a matching method is not found.

The method throws a `SecurityException` if access to the underlying method is denied.

See *The Java Language Specification*, sections 8.2 and 8.4.

getDeclaredConstructor

```
    public Constructor getDeclaredConstructor(Class[] parameterTypes)
        throws NoSuchMethodException, SecurityException
```

Returns a `Constructor` object that reflects the specified constructor of the class or interface represented by this `Class` object. The `parameterTypes` parameter is an array of `Class` objects that identify the constructor's formal parameter types, in declared order.

The method throws a `NoSuchMethodException` if a matching constructor is not found.

The method throws a `SecurityException` if access to the underlying constructor is denied.

Contents | Prev | Next

Copyright © 1996, 1997 Sun Microsystems, Inc. All rights reserved.

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- [✓] **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** ______________________

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.